OCFB: Output Ciphertext Feedback Mode

Authenticated Encryption Without a Block Cipher

Christian Forler[1] **Stefan Lucks**[1] **David McGrew**[2] **Jakob Wenzel**[1]

[1]Bauhaus-Universität Weimar, Germany

[2]Cisco Systems, USA

ESC 2013, Mondorf-les-Bains,
January, 2013

Outlook

Goals

1. Security (of course)
2. Feasible on constrained devices

   one primitive to rule them all,
   one primitive to bind them . . .
3. Simplicity:
   - easy to describe
   - easy to implement
   - easy to analyze

   based on a “standard” primitive
4. Reasonable efficiency

Goals

1. Security (of course)
2. Feasible on constrained devices

one primitive to rule them all,
one primitive to bind them . . .

3. Simplicity:
   - easy to describe
   - easy to implement
   - easy to analyze

   based on a “standard” primitive
4. Reasonable efficiency

Goals

1. Security (of course)
2. Feasible on constrained devices

one primitive to rule them all,
one primitive to bind them . . .

3. Simplicity:
   - easy to describe
   - easy to implement
   - easy to analyze

   based on a “standard” primitive
4. Reasonable efficiency

Goals

1. Security (of course)
2. Feasible on constrained devices

one primitive to rule them all,
one primitive to bind them . . .

3. Simplicity:
   - easy to describe
   - easy to implement
   - easy to analyze

   based on a “standard” primitive
4. Reasonable efficiency

Goals

1. Security (of course)
2. Feasible on constrained devices

one primitive to rule them all,
one primitive to bind them . . .

3. Simplicity:
   - easy to describe
   - easy to implement
   - easy to analyze

   based on a “standard” primitive
4. Reasonable efficiency

Goals

1. Security (of course)
2. Feasible on constrained devices

one primitive to rule them all,
one primitive to bind them . . .

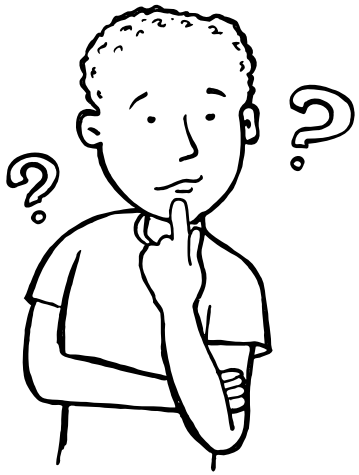
3. Simplicity:
   - easy to describe
   - easy to implement
   - easy to analyze

   based on a “standard” primitive
4. Reasonable efficiency

Requirements

Secure

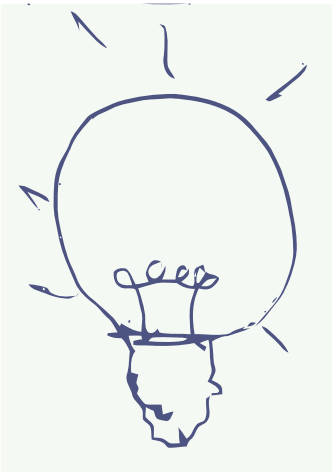
Simple



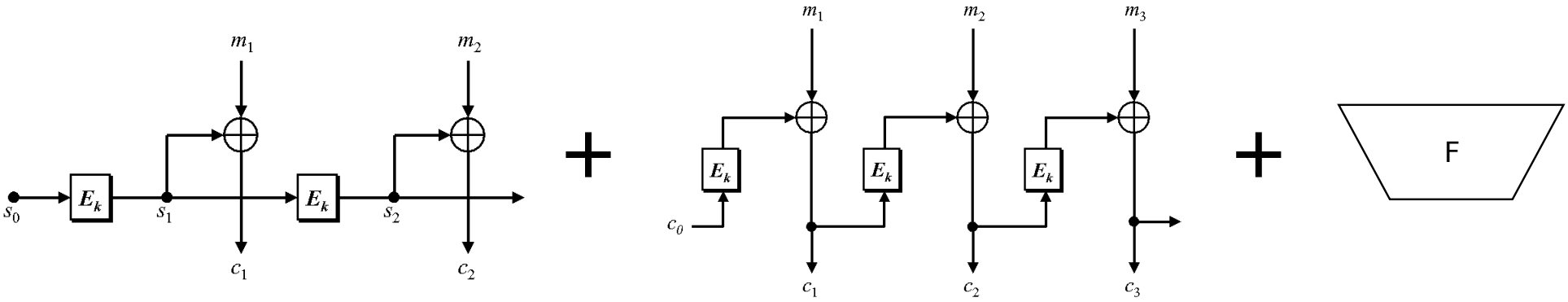
Feasible on constrained devices

Reasonable efficient

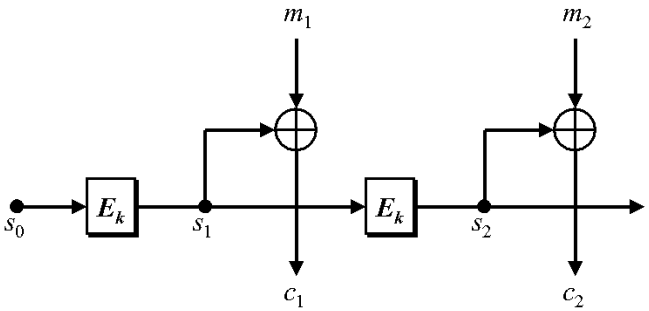
Enlightenment



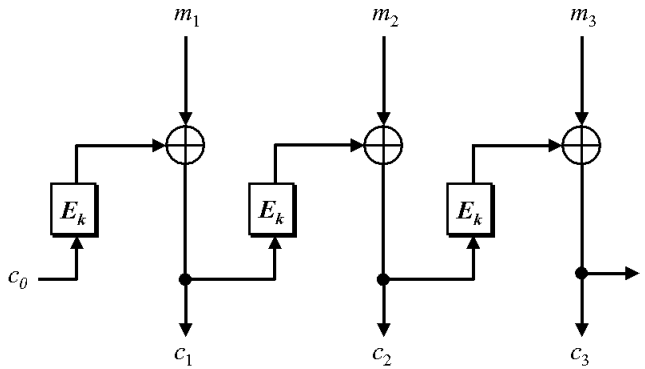
Design



Design


m_1
m_2
E_k
E_k
s_0
s_1
s_2
c_1
c_2

+


m_1
m_2
m_3
E_k
E_k
E_k
c_0
c_1
c_2
c_3

+


F

+

+

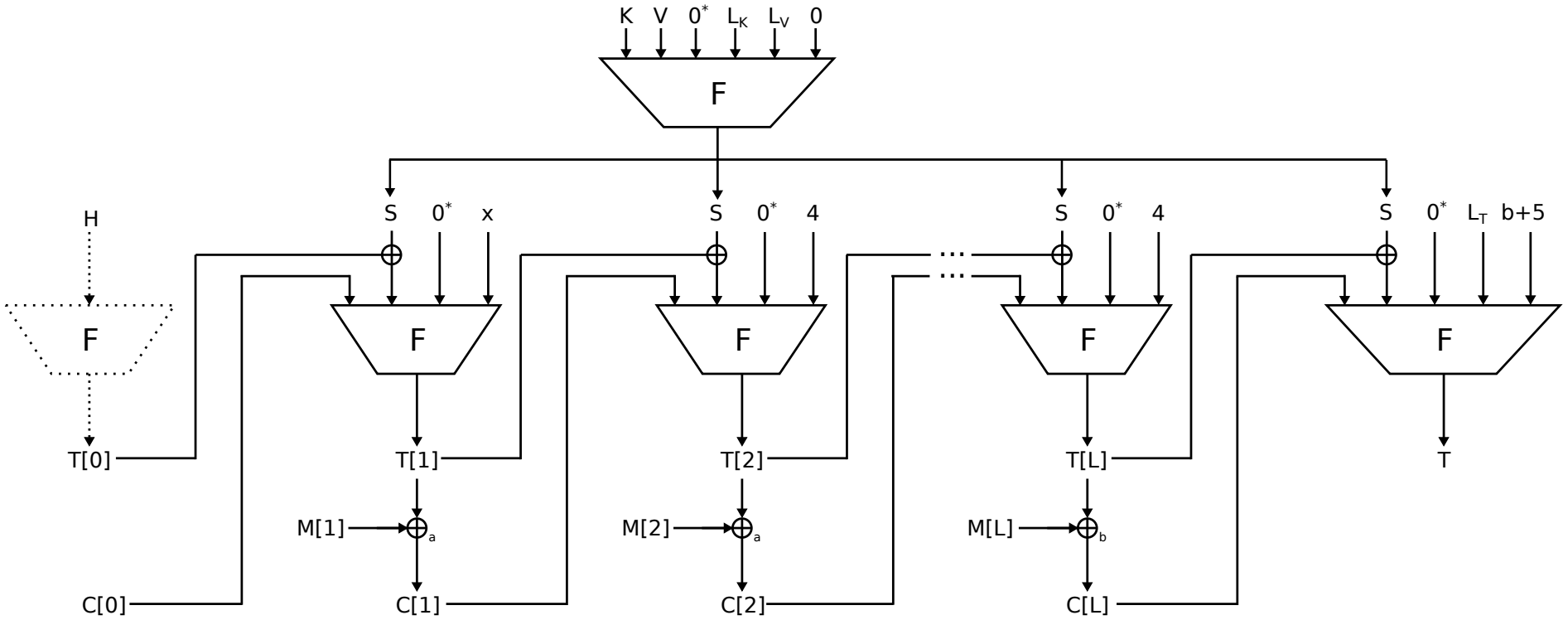
+

+


Put("Hello World");

=

OCFB


K V 0* L_K L_V 0
F
H
S 0* x
S 0* 4
S 0* 4
S 0* L_T b+5
F
F
F
F
F
T[0]
T[1]
T[2]
T[L]
T
M[1]
M[2]
M[L]
a
a
b
C[0]
C[1]
C[2]
C[L]

Block Cipher Based AE and Hashing

- We have plenty of good DBL hash functions, but
- what is the “standard” for DBL hashing?
- Digital signature standards require hash functions

$\Rightarrow$ The abandonment of block cipher is more appealing then the abandonment of a hash function.

Generic Composition

- Maintaining **two** keys for encryption scheme and MAC
- Memory is a very precious resource on constrained devices
- Slower than single pass solutions

Compression Function Based AE

- More efficient than hash function based design
- Typical standards do not formally define compression functions
- API must support compression function access
  (black box access is no longer sufficient)

Replacing the Block Cipher by a Hash Function

- Simple for key stream based AE schemes like CCFB or GCM
- A new design allows to
  - improve performance and
  - provide advanced security features like misuse resistance

Design Choices

- Second line of defence (misuse resistance)
- Side channel resistance
- Domain separation
- One compression function call per hash function invocation
  - Reasonable performance
  - Should apply for SHA-224/256, where

$$M' = M \,||\, 10^* \,||\, \mathtt{bitlen}_{64}(M) \qquad |M'| = 512,\ |M| \leq 447$$

  - $\Rightarrow$ Message size is less then twice the output size ($\frac{|M|}{[224|256]} < 2$)

Design Choices

- Second line of defence (misuse resistance)
- Side channel resistance
- Domain separation
- One compression function call per hash function invocation
  - Reasonable performance
  - Should apply for SHA-224/256, where

$$M' = M \,||\, 10^* \,||\, \texttt{bitlen}_{64}(M) \qquad |M'| = 512,\ |M| \leq 447$$

  - $\Rightarrow$ Message size is less then twice the output size ($\frac{|M|}{[224|256]} < 2$)

Design Choices

- Second line of defence (misuse resistance)
- Side channel resistance
- Domain separation
- One compression function call per hash function invocation
  - Reasonable performance
  - Should apply for SHA-224/256, where

$$M' = M \,||\, 10^* \,||\, \mathtt{bitlen}_{64}(M) \qquad |M'| = 512,\ |M| \leq 447$$

  - $\Rightarrow$ Message size is less then twice the output size ($\frac{|M|}{[224|256]} < 2$)

Design Choices

- Second line of defence (misuse resistance)
- Side channel resistance
- Domain separation
- One compression function call per hash function invocation
  - Reasonable performance
  - Should apply for SHA-224/256, where

$$M' = M \,||\, 10^* \,||\, \mathtt{bitlen}_{64}(M) \qquad |M'| = 512,\ |M| \leq 447$$

  - $\Rightarrow$ Message size is less then twice the output size ($\frac{|M|}{[224|256]} < 2$)

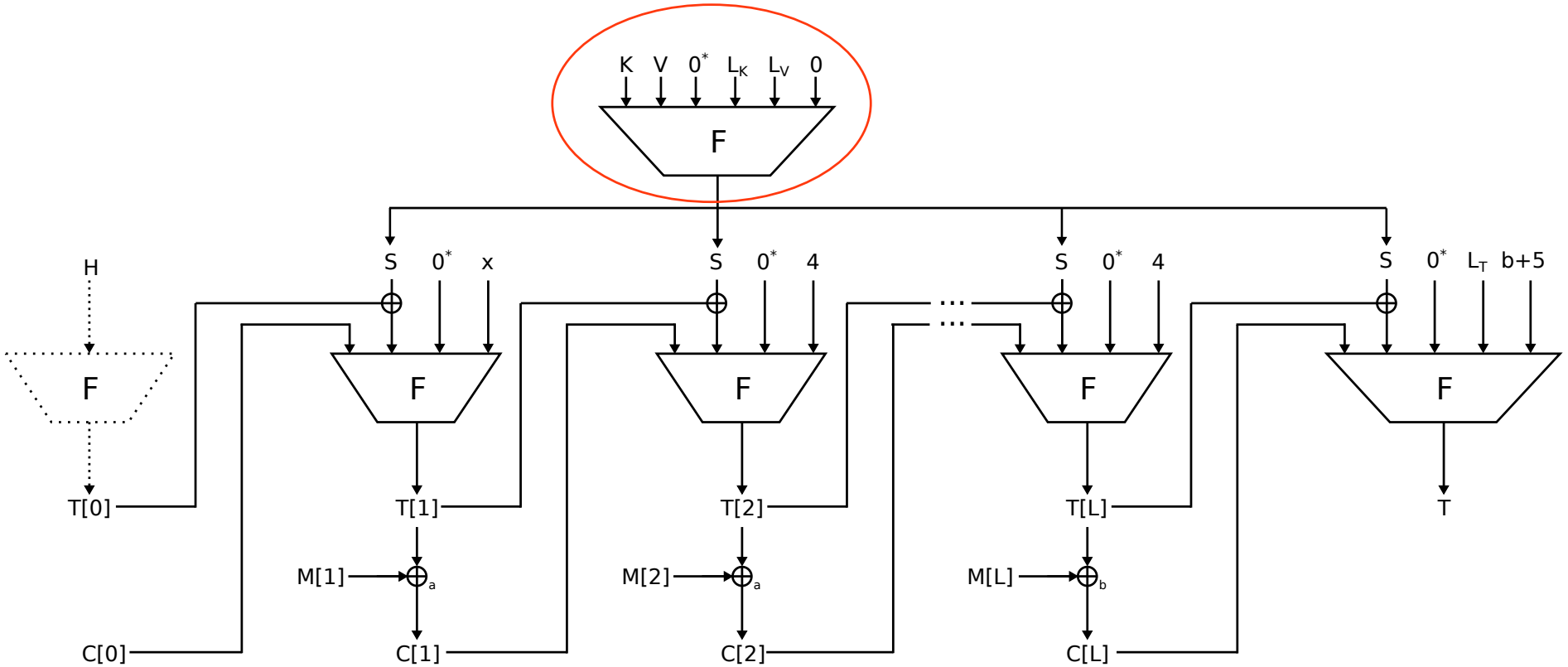
Design Choices

- Second line of defence (misuse resistance)
- Side channel resistance
- Domain separation
- One compression function call per hash function invocation
  - Reasonable performance
  - Should apply for SHA-224/256, where

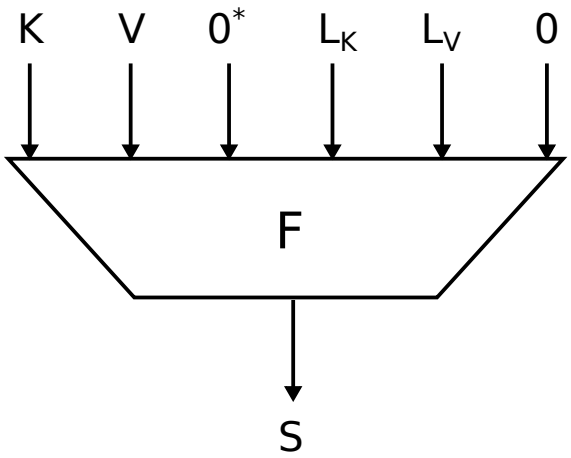
$$M' = M \,||\, 10^* \,||\, \mathtt{bitlen}_{64}(M) \qquad |M'| = 512,\ |M| \leq 447$$

  - $\Rightarrow$ Message size is less then twice the output size ($\frac{|M|}{[224|256]} < 2$)

Session Key Derivation



Session Key Derivation



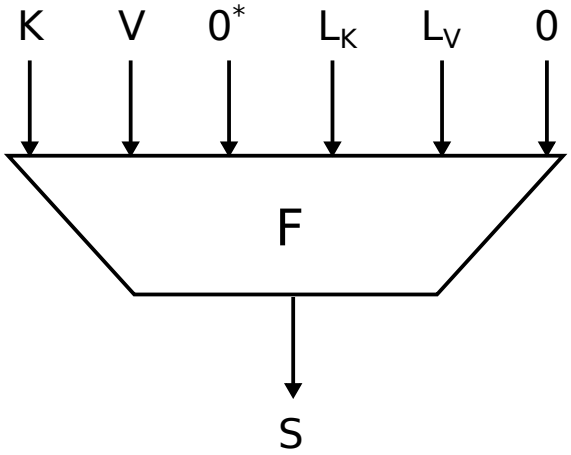
Session key S is derived from

- a long term key K (with $|K| = L_K \geq n$) , and
- a nonce V

Why?

- Inexpensive due to the absence of a key scheduler :-)
- Provides some protection against side-channel attacks ($\rightarrow$ security claims)

Session Key Derivation



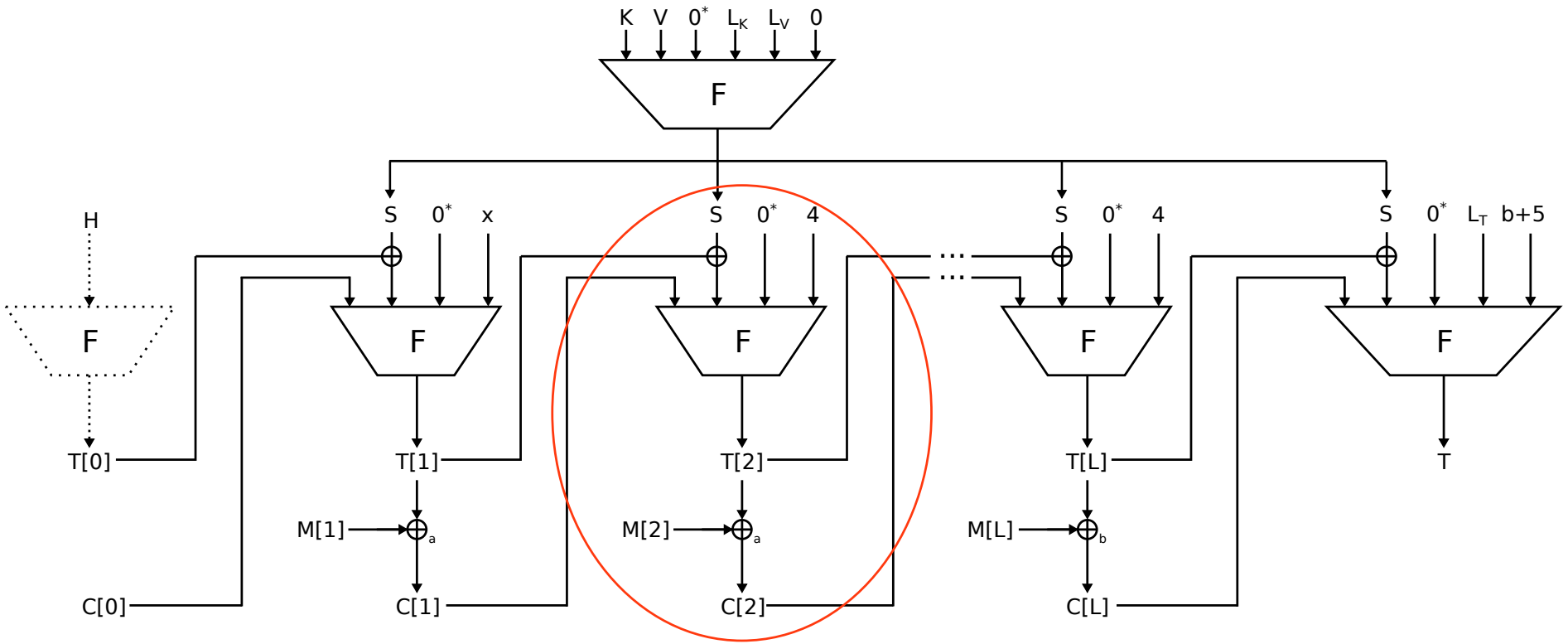
Session key S is derived from

- a long term key K (with $|K| = L_K \geq n$) , and
- a nonce V

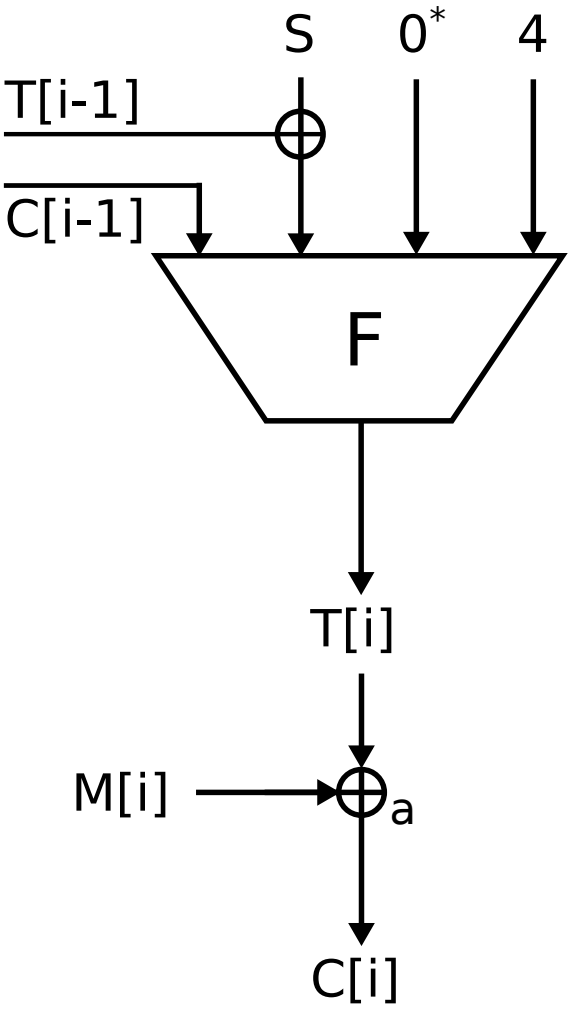
Why?

- Inexpensive due to the absence of a key scheduler :-)
- Provides some protection against side-channel attacks ($\rightarrow$ security claims)

Message Block Processing

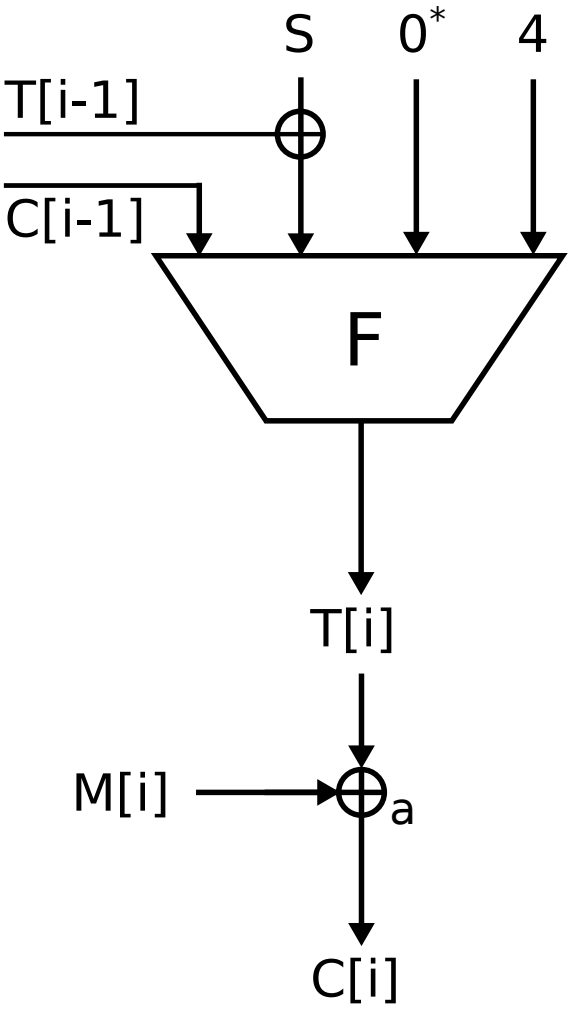


Message Block Processing



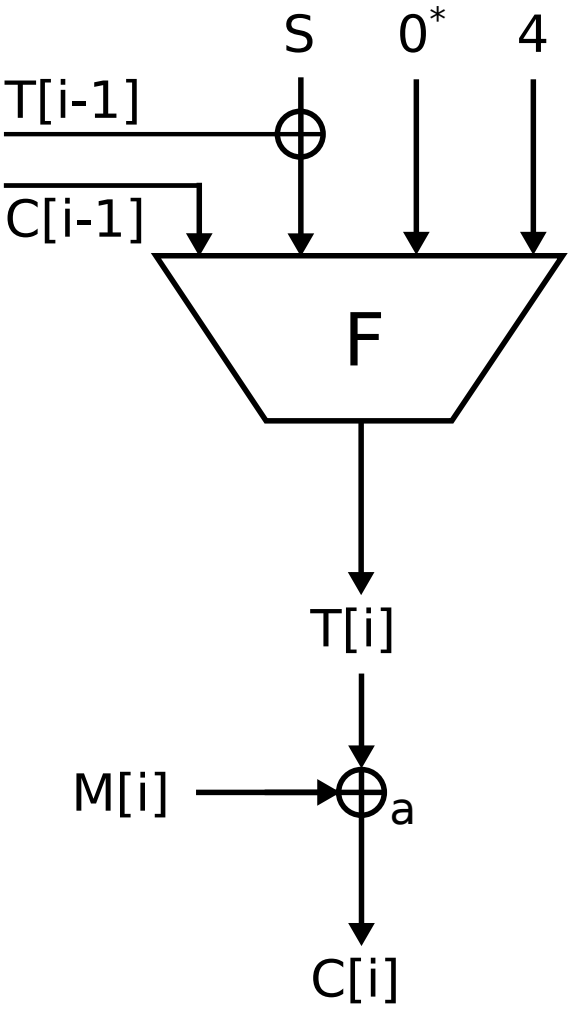
- Why $T[i-1] \oplus S$ instead of $T[i-1] \,||\, S$?
- SHA-224: $\texttt{bitlen}(T[i-1] \,||\, S) > 447$
- Recommended message block length
  - SHA-224: $a = 192$
  - SHA-256: $a = 160$

Message Block Processing



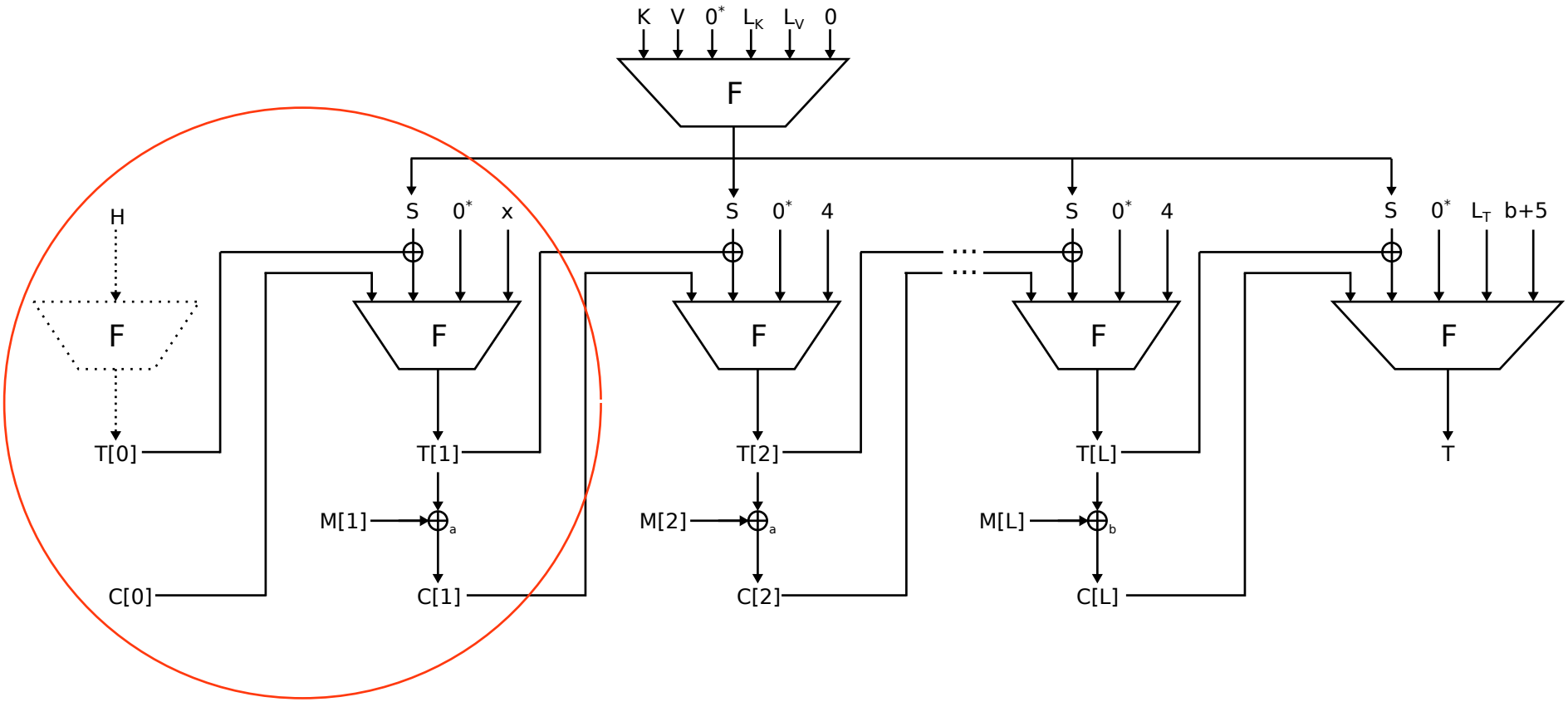
- Why $T[i-1] \oplus S$ instead of $T[i-1] \,||\, S$?
- SHA-224: $\texttt{bitlen}(T[i-1] \,||\, S) > 447$
- Recommended message block length
  - SHA-224: $a = 192$
  - SHA-256: $a = 160$

Message Block Processing

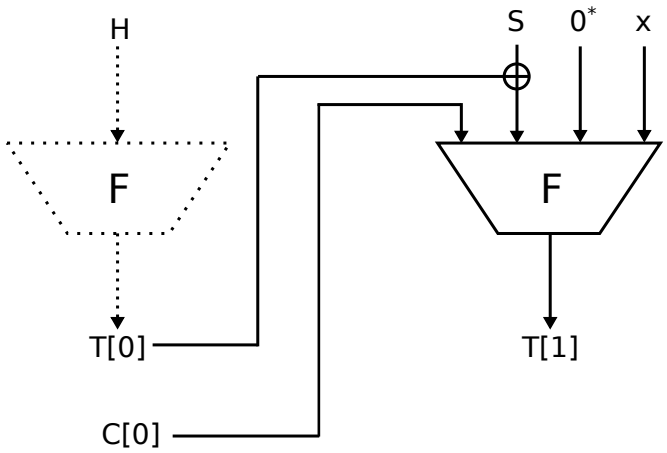


- Why $T[i-1] \oplus S$ instead of $T[i-1] \,||\, S$?
- SHA-224: $\texttt{bitlen}(T[i-1] \,||\, S) > 447$
- Recommended message block length
  - SHA-224: $a = 192$
  - SHA-256: $a = 160$

Associated Data Processing



Associated Data Processing

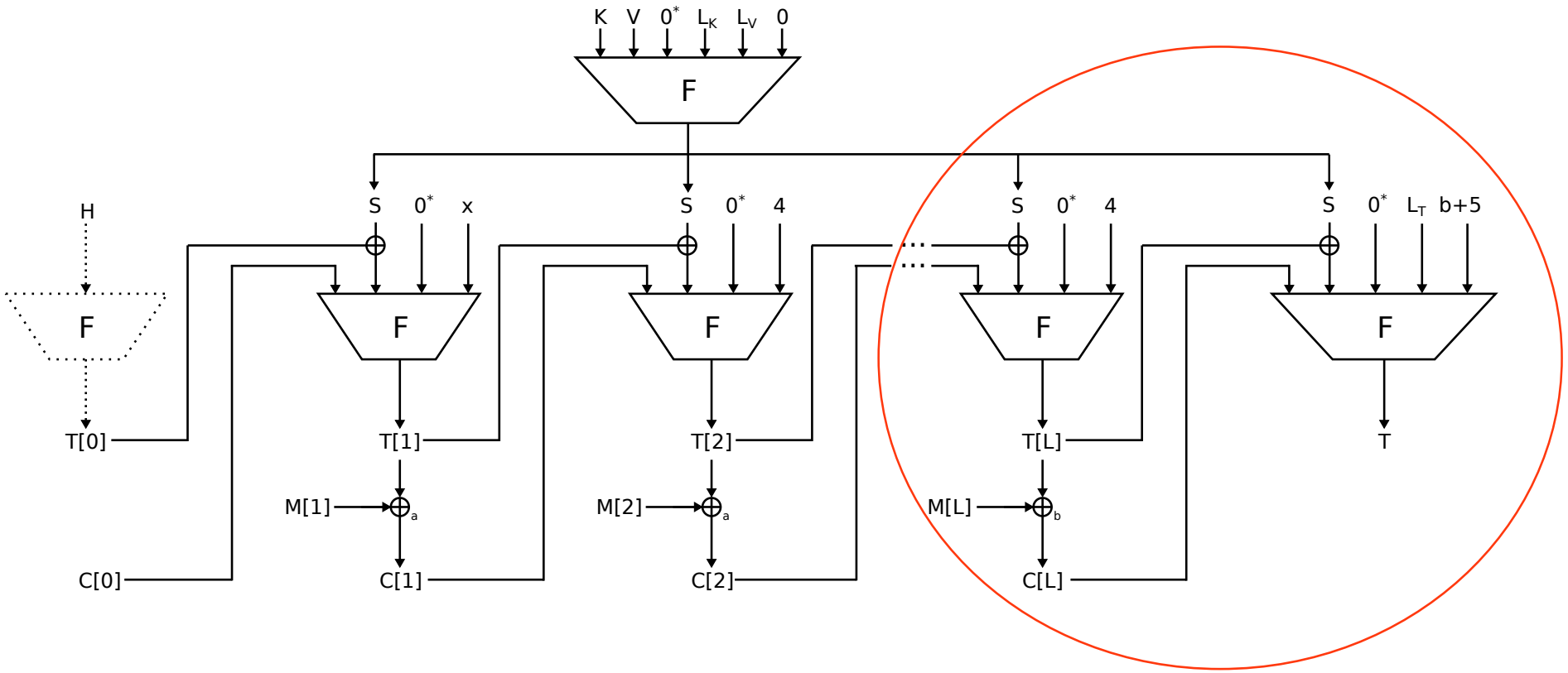


Omitting invocation of F, if possible

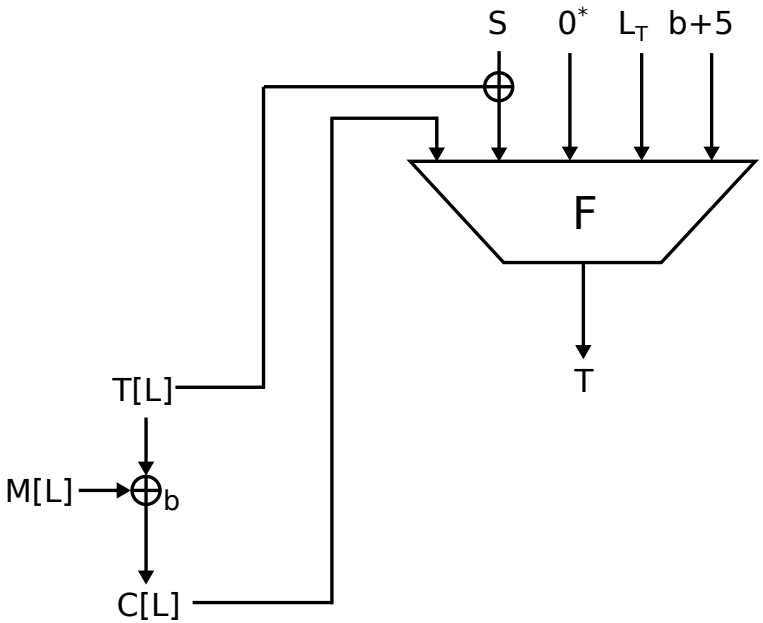
$$(x, T[0]) := \begin{cases} \texttt{0x01}, H \,||\, 10^* & \text{if } |H| < n, \\ \texttt{0x02}, H & \text{if } |H| = n, \\ \texttt{0x03}, F(H) & \text{else.} \end{cases}$$

$$C[0] = 0x1415926\ldots$$

Authentication Tag Generation



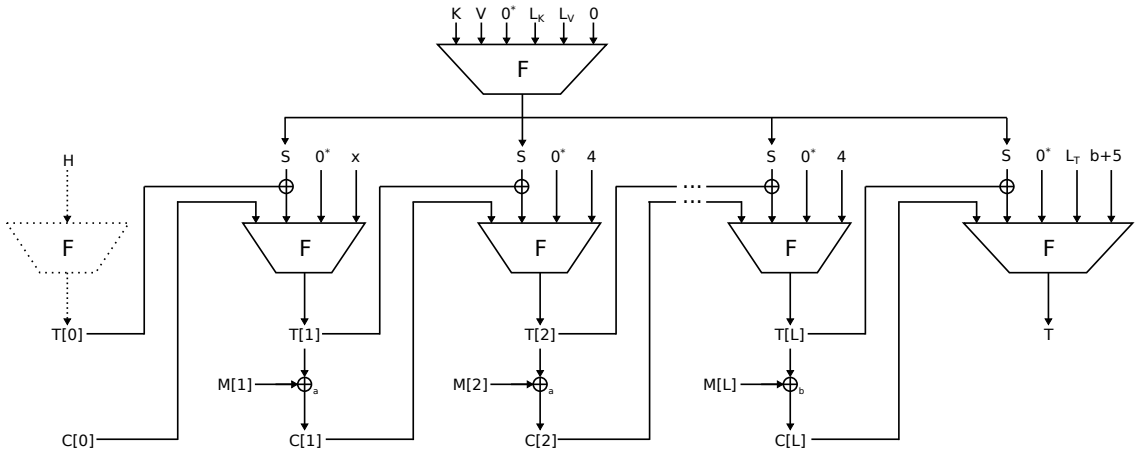
Authentication Tag Generation



- Domain is derived from the bit length of the last message block (b)
- Tag length T_L can be set up to n

Security

Standard AE

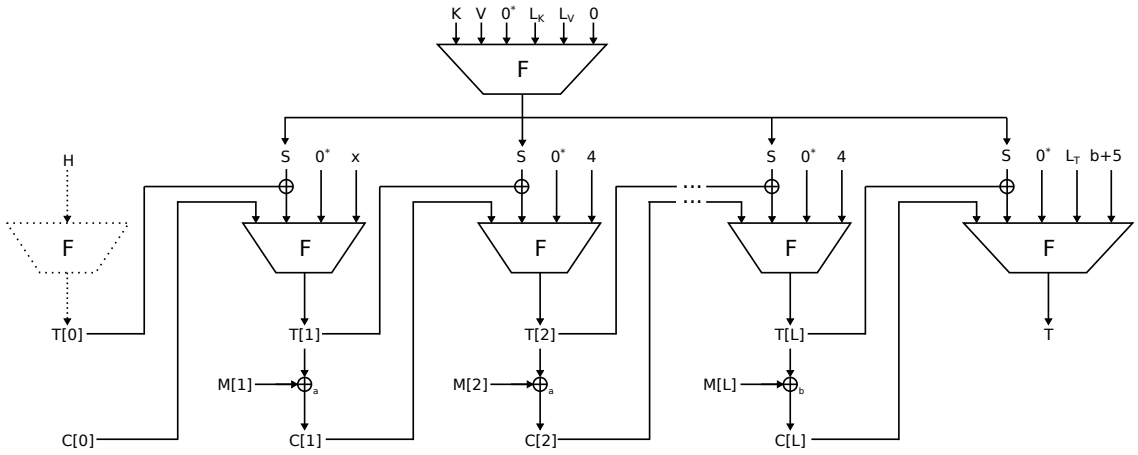


- Assume the hash function behaves like a good PRF
- Restrict the adversary to be *nonce-respecting*
- Mark input and output collisions as **bad events**

- **privacy:** chosen plaintext attack (CPA) resistant
- **authenticity:** integrity of ciphertexts (Int-CTXT)

Security

Standard AE

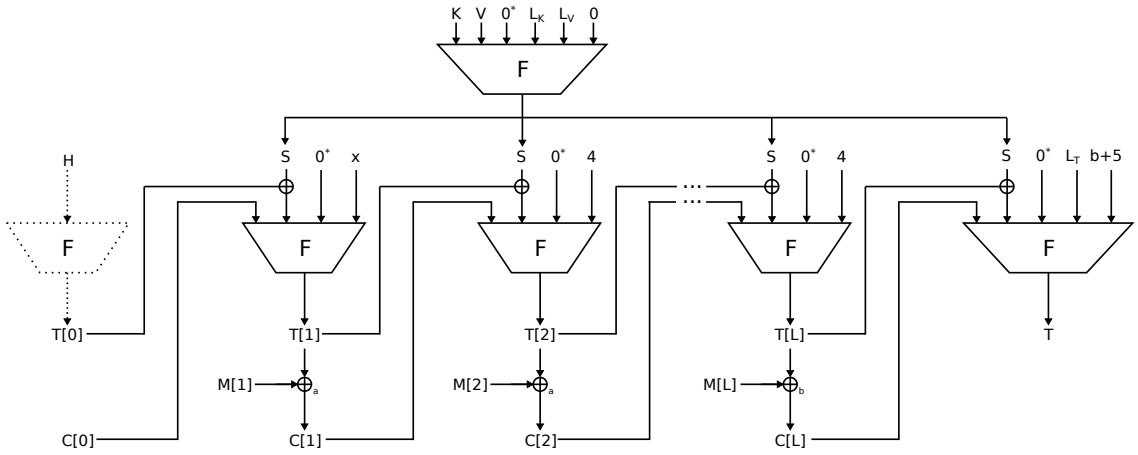


- Assume the hash function behaves like a good PRF
- Restrict the adversary to be *nonce-respecting*
- Mark input and output collisions as **bad events**

- **privacy:** chosen plaintext attack (CPA) resistant
- **authenticity:** integrity of ciphertexts (Int-CTXT)

Security

Standard AE

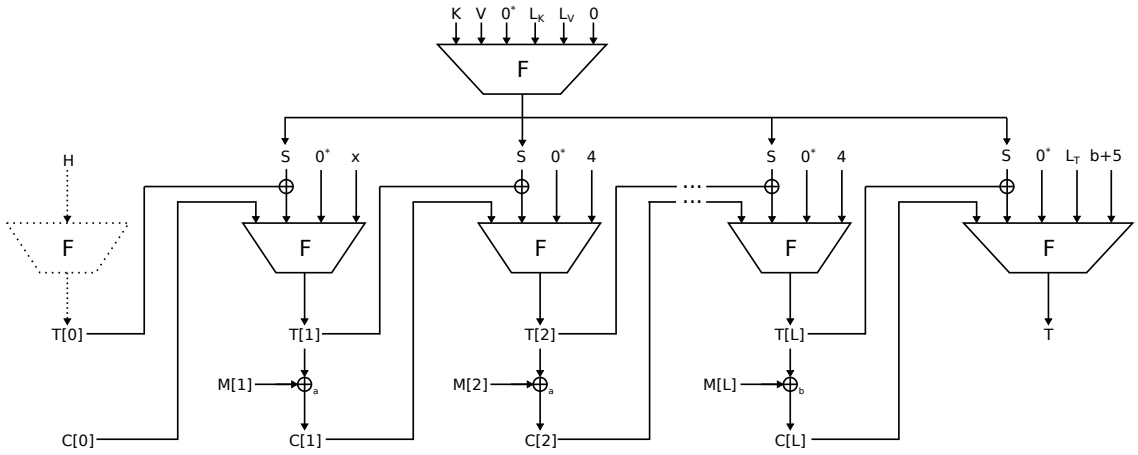


- Assume the hash function behaves like a good PRF
- Restrict the adversary to be *nonce-respecting*
- Mark input and output collisions as **bad events**

- **privacy:** chosen plaintext attack (CPA) resistant
- **authenticity:** integrity of ciphertexts (Int-CTXT)

Security

Standard AE



- Assume the hash function behaves like a good PRF
- Restrict the adversary to be *nonce-respecting*
- Mark input and output collisions as **bad events**

- **privacy:** chosen plaintext attack (CPA) resistant
- **authenticity:** integrity of ciphertexts (Int-CTXT)

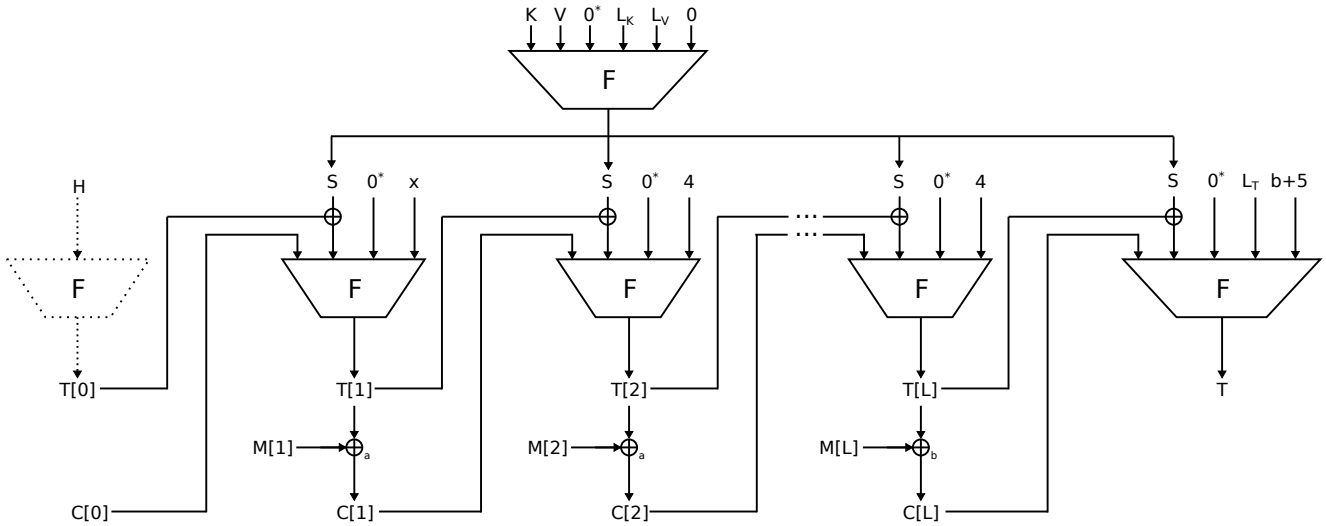
CCA3-Security

$$\textbf{CCA3-Adv}(q, \ell, t) \leq \frac{7\ell^2+5q^2}{2^n} + \frac{q}{2^{L_T}} + 2 \cdot \textbf{Adv}_F^{\text{PRF-RK}}(q+\ell, O(t))$$

- L_T denotes the tag length in bits
- q denotes the number of OCFB queries, and
- ℓ denotes the number of compression function calls.

Security

Nonce Misuse Resistance

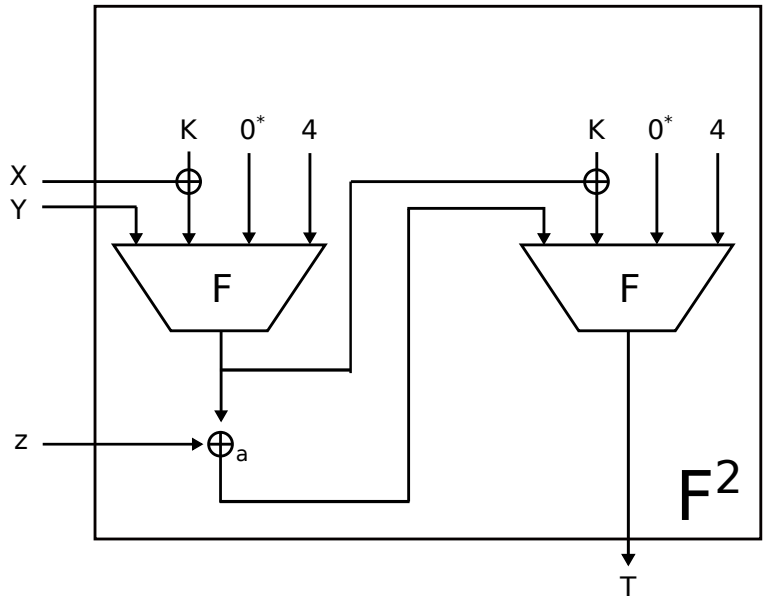


nonce misuse: the adversary is not always *nonce respecting* (*e.g.,* due to implementation errors)

- **privacy:** still holds when using a new nonce
- **authenticity:** not affected, due to the OPerm structure (!)

Security

Forgery Resistance Assumption



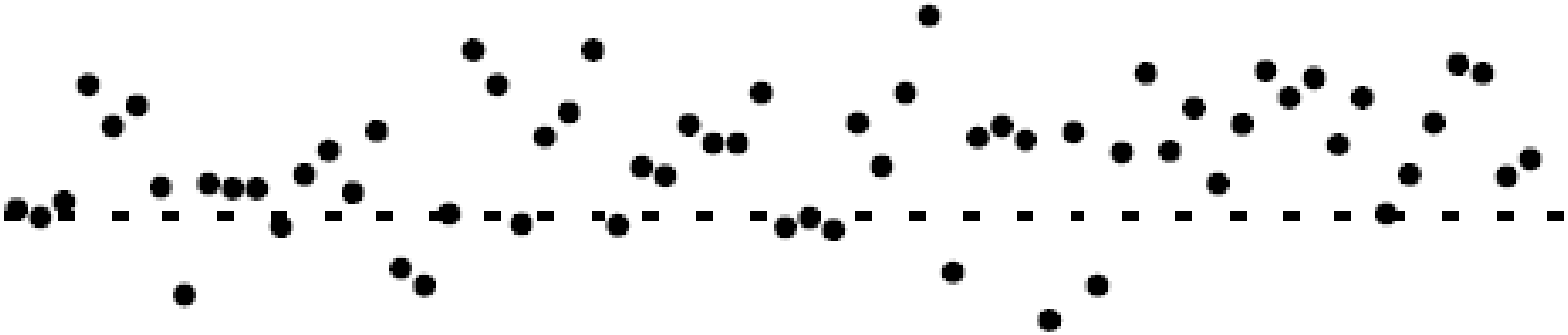
weak assumptions:

- **privacy:** requires the F to be a good PRF
- **authenticity:** only requires "forgery resistance" of F^2

Security Claims

. . . on Side Channels

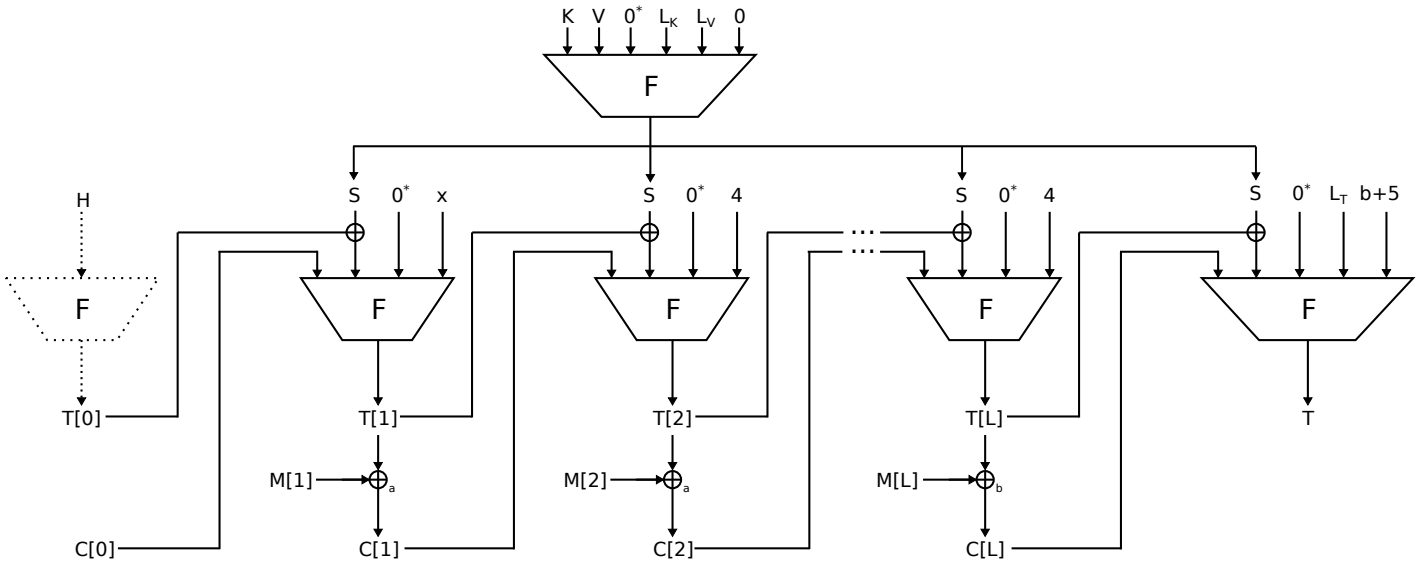
Typical side-channel attacks:



- Many measurements of a primitive operations under the same key
- X messages, each of length L blocks:
  XL measurements for the same key

Security Claims

. . . on Side Channels



Side-channel attacks against OCFB:

- X messages, each of length L, nonce-respecting:
  X measurements for **key** and L for each of **S**
- even when not nonce-respecting:
  adversary may find some **S** but only use it to to compromise messages using that single **nonce**

Final Remarks and Summary

- Our goals:
  - Secure, feasible on constrained devices, simple, efficient (in that order)
  - Using a hash function seems to be a good approach
- Security requirements (beyond “standard”):
  - Authenticity even under nonce reuse
  - Authenticity needs weaker assumption than privacy
  - Some defense against side-channel attacks

Should such security requirements become a standard for new generation AE schemes?

Final Remarks and Summary

- Our goals:
  - Secure, feasible on constrained devices, simple, efficient (in that order)
  - Using a hash function seems to be a good approach
- Security requirements (beyond “standard”):
  - Authenticity even under nonce reuse
  - Authenticity needs weaker assumption than privacy
  - Some defense against side-channel attacks

Should such security requirements become a standard for new generation AE schemes?

Questions?